CSCI-101 Programming 1
Lab 17

The following program allows the user to *walk* around a maze. The users current location in the maze is indicated by an 'X' character.

INSTRUCTIONS

Create directory named **lab17**. Then copy your **Cell.java** and your **maze.txt** file from your **lab16** directory into your **lab17** directory.

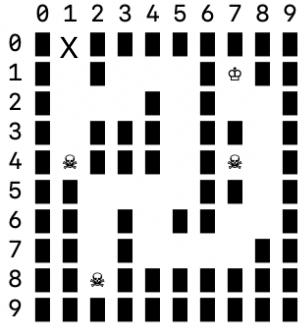
Create a file named **MazeGame.java** that includes the following:

1. Add the following static fields to the class:
   ```
   static int numRows = 0;
   static int numCols = 0;
   static Cell[][] maze = null;
   static int playerRow = 9;
   static int playerCol = 9;
   ```

2. Add a method named **loadMaze** that does the following:
   - Ask the user to enter a file name and reads the file name from the keyboard.
   - Create a Scanner that can read from the file.
   - Configure the Scanner to use commas and newline characters as delimiters.
   - Read the number of rows and columns from the file and stores their values in the static fields named **numRows** and **numCols**, respectively.
   - Initialize the static field named **maze** to a 2D array of **Cell** elements having the same number of rows and columns as the values in **numRows** and **numCols**, respectively.
   - Read the rest of the values in the file and for each subsequent integer read from the file, set the corresponding element in **maze** equal to a new a new **Cell** element whose **code** field is equal to the value read from the file.

3. Add a method named **printMaze**. The method prints to the screen the row and column coordinates, the maze glyphs, and the users current location indicated by an X, similar to what is shown below.

   Note: The players current location is specified in the static fields **playerRow** and **playerCol**.



4. Add a method named **isValidDirection**. The method has one parameters named **direction** which holds a character.

   If the value of **direction** is a character other than ‘w’, ‘a’, ’s’, or ‘d’, then return **false**.

   Otherwise **direction** is either 'w' (up), 'a' (left), 's' (down) or 'd' (right) and indicates where the player would like to move.

   Note: The players current location is specified in the static fields **playerRow** and **playerCol**.

   If the location of the player's next move doesn't take the player off the path in the **maze** then return **true**; otherwise return **false**.

5. Add a method named **updateUser**. The method has one parameter named **direction** which holds a character.

   Note: The players current location is specified in the static fields **playerRow** and **playerCol**.

   This method assumes **direction** is a valid direction for the user to move and “moves” the user by updating the value of **playerRow** and/or **playerCol** based on the value of **direction**.

6. In the **main** method do the following.
   1. Declare the following variables:

      ```
      String input = null;
      char choice = '?';
      boolean isValid = false;
      ```

   2. Create a Scanner that can read data from the keyboard.

   3. Initialize the static 2D array named **maze** by calling **loadMaze**.

   4. Traverse the perimeter of the maze to find a **Cell** that is blank. Consider that to be the users starting point. Set the static fields **playerRow** and **playerCol** equal to the row and column of the **Cell** at which you wish to start the player.

   5. Add a do-while loop that executes a block of code while the value in the variable named **choice** is not 'x'.

      ```
      do {



      } while (choice != 'x');
      ```

   6. Inside the **do-while** loop do the following:

a. Print to the screen the maze by calling **printMaze**.

b. Ask the user to enter a direction: w (up), a (left), s (down), d (right) or x (to exit)

c. Read the user's choice from the keyboard and store the value in the variable named **input**.

d. If the length of the **input** string entered by the user is 0,
   - then use a **continue;** statement to continue execution at the top of the loop.

   else,
   - set **choice** equal to the first character in the string entered by the user and
   - set **isValid** equal to the value returned by **isValidDirection** when passed **choice.**

e. If the direction is valid (i.e. **isValid** is true), call **updateUser**.

f. Reset the screen using the following code:

```
System.out.print("\033[H\033[2J");
System.out.flush();
```